ФПМИ БГУ

Вычислительные методы алгебры
Лаборатоная работа 2

Решение СЛАУ методом квадратного корня

Подготовил:
Ткачук Павел
2 курс 1 группа

Преподаватель:
Будник Анатолий Михайлович

21 декабря 2016 г.

1 Постановка задачи

Система:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n = b_n \end{cases}, \ a_{ij} = a_{ji}, \ \forall i, j \tag{1}$$

Входные данные:

$$\left[\begin{array}{ccccc|c} 0.6444 & 0.0000 & -0.1683 & 0.1184 & 0.1973 & 1.2677 \\ -0.0395 & 0.4208 & 0.0000 & -0.0802 & 0.0263 & 1.6819 \\ 0.0132 & -0.1184 & 0.7627 & 0.0145 & 0.0460 & -2.3657 \\ 0.0395 & 0.0000 & -0.0960 & 0.7627 & 0.0000 & -6.5369 \\ 0.0263 & -0.0395 & 0.1907 & -0.0158 & 0.5523 & 2.8351 \end{array}\right]$$

Домнажаем слева матричное уравнение на матрицу A^T чтобыл получить симметрическую систему

$$\left[\begin{array}{ccccc|c} 0.4965221 & -0.02976049 & -0.10906373 & 0.13191428 & 0.09195098 & 0.53559917 \\ -0.02976049 & 0.18575662 & -0.05029722 & -0.06272879 & -0.0018678 & 0.87585595 \\ -0.10906373 & -0.05029722 & 0.59823034 & -0.06163865 & 0.17564755 & -0.84947733 \\ 0.13191428 & -0.06272879 & -0.06163865 & 0.59248754 & -0.02931901 & -5.04958356 \\ 0.09195098 & -0.0018678 & 0.17564755 & -0.02931901 & 0.34390336 & 1.75135471 \end{array}\right]$$

Задача:

1. Методом квадратного корня найти решение СЛАУ x
2. Найти вектор невязки $r = Ax - b$

2 Алгоритм

1. Приводим матрицу A к виду $A = S^T S$

$$\begin{cases} s_{ij} = 0, \ j = \overline{1, i-1} \\ s_{ii} = \sqrt{\left| a_{ii} - \sum\limits_{k=1}^{i-1} |s_{kj}|^2 \right|}, \\ s_{ij} = \dfrac{a_{ij} - \sum\limits_{k=1}^{i-1} s_{ki}s_{kj}}{s_{ii}}, \ j = \overline{i+1, n} \end{cases} \qquad i = 1, 2, \ldots, n$$

2. Решаем систмемы с треугольными матрицами

$$\begin{cases} S^T y = f, \\ Sx = y \end{cases}$$

$$\begin{cases} y_i = \dfrac{f_i - \sum\limits_{k=1}^{i-1} s_{ki}y_k}{s_{ii}}, \qquad i = \overline{1, n} \\ x_j = \dfrac{y_j - \sum\limits_{k=j+1}^{n} s_{jk}x_k}{s_{jj}}, \quad j = \overline{n, 1}. \end{cases}$$

3 Результаты и вывод

3.1 Входные данные

0.6444 0.0000 -0.1683 0.1184 0.1973 1.2677
-0.0395 0.4208 0.0000 -0.0802 0.0263 1.6819

0.0132 -0.1184 0.7627 0.0145 0.0460 -2.3657
0.0395 0.0000 -0.0960 0.7627 0.0000 -6.5369
0.0263 -0.0395 0.1907 -0.0158 0.5523 2.8351

3.2 Выходные данные

```
Решаем матричное уравнение методом квадратного корня
Основная матрица системы:
[[ 0.41923779 -0.01922333 -0.09716147  0.10936737  0.14123396]
 [-0.01922333  0.19265145 -0.09783633 -0.03484086 -0.01619521]
 [-0.09716147 -0.09783633  0.65561867 -0.08509983  0.10720222]
 [ 0.10936737 -0.03484086 -0.08509983  0.60262178  0.01319172]
 [ 0.14123396 -0.01619521  0.10720222  0.01319172  0.34677027]]
Свободные члены:
[ 0.53559917  0.87585595 -0.84947733 -5.04958356  1.75135471]
Решение системы:
[ 0.99821505  1.99986528 -2.99975971 -9.00000843  6.00705353]
Невязка:
2.71947991102e-16
```

3.3 Вывод

Рассмотрим ответ, полученный методом Гаусса:

```
[0.99821505  1.99986528 -2.99975971 -9.00000843  6.00705353]
```

Ответы полученные методом квадратного корня в точности совпадают с ответами метода Гаусса. Как и в методе Гаусса, координаты вектора невязки достаточно малы, что говорит о том, что найденный ответ достаточно близок к точному.

4 Листинг кода

```
import numpy as np
import numpy.linalg as linalg


# Метод квадратного корня


def solve(A, b):
    size = len(b)
    S = np.zeros((size, size))
    for i in range(size):
        S[i, i] = np.sqrt(A[i, i] - sum(([(S[k, i] ** 2) for k in range(i)])))
        for j in range(i + 1, size):
            S[i, j] = (A[i, j] - sum([S[k, i] * S[k, j] for k in range(i)])) / S[i, i]
    y = np.zeros(size)
    for i in range(size):
        y[i] = (b[i] - sum([S[k, i] * y[k] for k in range(i)])) / S[i, i]
    x = np.zeros(size)
    for i in reversed(range(size)):
        x[i] = (y[i] - sum(S[i, k] * x[k] for k in range(i + 1, size))) / S[i, i]
    return x


# Основная программа

file = open("matrix", "r")  # Чтение файла
A, b = [], []
for line in file:
    A.append([float(el) for el in line.split()[:-1]])
    b.append(float(line.split().pop()))
A = np.array(A)
b = np.array(b)
b = np.dot(np.transpose(A), b)
A = np.dot(np.transpose(A), A)
print("Решаем матричное уравнение методом квадратного корня")
print("Основная матрица системы:")
print(A)
print("Свободные члены:")
print(b)
ans = solve(A, b)
print("Решение системы:")
print(ans)
print("Невязка:")
print(linalg.norm(np.dot(A, ans) - b))
```